NPS ARCHIVE
2000
DEBRINE, J.

DUL
NAV GRADUAT HOOL
MON CA 9394 01

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

RE-PURPOSING COMMERCIAL ENTERTAINMENT SOFTWARE FOR MILITARY USE

By

Jeffrey D. DeBrine
Donald E. Morrow

September 2000

Thesis Advisor: Michael Capps
Co-Advisor: Michael Zyda

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2000	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE
Re-Purposing Commercial Entertainment Software for Military Use

5. FUNDING NUMBERS
MIPROEMANPGS00

6. AUTHOR(S)
DeBrine, Jeffrey D. and Morrow, Donald E.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Economic & Manpower Analysis
607 Cullum Rd, Floor 1B, Rm B109, West Point, NY 10996-1798

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(maximum 200 words)***

Virtual environments have achieved widespread use in the military in applications such as theater planning, training, and architectural walkthroughs. These applications are generally expensive and inflexible in design and implementation. Re-purposing these applications to meet the dynamic modeling and simulation needs of the military can be awkward or impossible.

Video games are designed to be both technologically advanced and flexible in design. We evaluated current games and modified Quake 3 Arena™ (Q3A) to serve as both an architectural walkthrough and a primitive team trainer. To accomplish this, we incorporated a real Naval Postgraduate School building into Q3A. We also modified the game's source code, characters and their behaviors, weapons models and characteristics, and overall gameplay.

By re-purposing commercial entertainment software, we have produced a viable military virtual environment application that is less expensive yet arguably as engaging as current computer-based options. This application was created in approximately 300 man-hours with a cost of $6780 (including hardware)—far less than the development time and cost of similar military virtual environment applications. Game evaluations included in this thesis facilitate and inform similar modification efforts by highlighting entertainment technology available in the year 2000 game market.

14. SUBJECT TERMS: Modeling and Simulation, Software Re-Purposing, Video Games, Entertainment Technology, Architectural Walkthrough, Game Modification

15. NUMBER OF PAGES
164

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFI- CATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239- 18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

RE-PURPOSING COMMERCIAL ENTERTAINMENT SOFTWARE FOR MILITARY USE

Jeffrey D. DeBrine
Lieutenant, United States Navy
B.S., United States Naval Academy, 1992

Donald E. Morrow
Lieutenant, United States Navy
B.S., Auburn University, 1992

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN
MODELING, VIRTUAL ENVIRONMENTS AND SIMULATION

from the

NAVAL POSTGRADUATE SCHOOL
September 2000

NPS Archive
2000
Debrine, J.

~~Thesis
D18925
C.1~~

THIS PAGE INTENTIONALLY LEFT BLANK

DUD[illegible] OX LIBRARY
NAV[illegible] GRADUAT[illegible] HOOL
MON[illegible] CA 9394[illegible] 01

ABSTRACT

Virtual environments have achieved widespread use in the military in applications such as theater planning, training, and architectural walkthroughs. These applications are generally expensive and inflexible in design and implementation. Re-purposing these applications to meet the dynamic modeling and simulation needs of the military can be awkward or impossible.

Video games are designed to be both technologically advanced and flexible in design. We evaluated current games and modified Quake 3 Arena™ (Q3A) to serve as both an architectural walkthrough and a primitive team trainer. To accomplish this, we incorporated a real Naval Postgraduate School building into Q3A. We also modified the game's source code, characters and their behaviors, weapons models and characteristics, and overall gameplay.

By re-purposing commercial entertainment software, we have produced a viable military virtual environment application that is less expensive yet arguably as engaging as current computer-based options. This application was created in approximately 300 man-hours with a cost of $6780 (including hardware)—far less than the development time and cost of similar military virtual environment applications. Game evaluations included in this thesis facilitate and inform similar modification efforts by highlighting entertainment technology available in the year 2000 game market.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. **OVERVIEW** 1

A. THESIS STATEMENT 1

B. MOTIVATION 1

1. Military VE Training Applications 1
2. Architectural Walkthroughs 2
3. COTS Entertainment Software 3

C. APPROACH 3

D. THESIS ORGANIZATION 5

II. **BACKGROUND & RELATED WORK** 7

A. INTRODUCTION 7

B. INTRODUCTION TO GAME TECHNOLOGY 7

1. Hardware Platform 8
2. Game Engine 9
3. Application 10
4. Modifications 10

C. PREVIOUS MODIFICATIONS TO COTS SOFTWARE 11

1. DOOM Trainer 11
2. Multi-User Information Visualization 11
3. Unrealty 12
4. Land Warrior 13

D. TEAM DYNAMICS 13

1. Characteristics of Effective Group Members 14
2. Characteristics of Effective Leaders 15
3. Summary 16

III. **CURRENT GAME TECHNOLOGY EVALUATION** 17

A. INTRODUCTION 17

B. TEAMPLAY IN GAMES, DEFINITIONS 18

C. GAME EVALUATIONS 20

1. EverQuest™ 21
2. Asheron's Call™ 22
3. Ultima™ Online: The Second Age 22

4. Delta Force™ 2 23
5. Half-Life®: Opposing Force™ 24
6. Counter-Strike 24
7. Team Fortress Classic 25
8. Quake III Arena™ 26
9. Tom Clancy's Rainbow Six 27
10. Tom Clancy's Rainbow Six Rogue Spear™ 28
11. Soldier of Fortune® 28
12. Spec Ops™: Ranger Gold 28
13. Starsiege™ Tribes 29
14. SWAT™ 3 30
15. Thief™ Gold 30
16. Unreal™ Tournament 31
17. The Sims™ 31
18. Microsoft® Allegiance™ 32
19. Sammy Sosa High Heat Baseball 32
20. iM1A2 Abrams™ 33

D. EVALUATION OF GAME ENGINES 36

E. FINAL CANDIDATES FOR RE-PURPOSING 38

IV. PROOF OF CONCEPT 39

A. INTRODUCTION 39
1. Mod Development Tools 39
2. Why We Chose Quake III Arena™ 40

B. STEPS IN MOD CREATION 41
1. Scenario Design 41
2. Level Design 42
3. Modeling 43
4. Artificial Intelligence 44
5. Gameplay 45
6. User Interface 46

V. IMPLEMENTATION 47

A. HERRMANN HALL MODEL 47
1. Converter 47
2. QERadiant (Optimization and Texturing) 50

B. QUAKE PAK FILE FORMAT 53

C. ARTIFICIAL INTELLIGENCE 54

D. IN-GAME MODELS 60
  1. Laser Designator 60
  2. Weapons 63
  3. Player Skin 65

E. SOURCE CODE CHANGES 70
  1. Weapons Characteristics (Gauntlet, Railgun) 70
  2. Removing Ammunition From Weapons 72
  3. One Life in Gameplay 73
  4. In-Game Menu Changes 74

F. SUMMARY 76

VI. ANALYSIS 77

A. SUMMARY OF WORK 77

B. BENEFITS OF REPURPOSING 78
  1. High Fidelity and Fast Performance 78
  2. Development Time and Cost 78

C. CONCLUSIONS AND RECOMMENDATIONS 80

D. FUTURE WORK 82
  1. Implement Training Metrics 82
  2. Build a Team Trainer 82
  3. Build a Theater Planner 83
  4. Build a Game Evaluation Database 83

APPENDIX A. CONVERTER CODE 85

APPENDIX B. Q3A ARTIFICIAL INTELLIGENCE CODE 93

APPENDIX C. Q3A MENU CODE 115

LIST OF REFERENCES 125

INITIAL DISTRIBUTION LIST 127

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1. Level of Effort to Modify an Existing Game 8
Figure 2. AutoCAD® versus Quake® Plane Definitions 48
Figure 3. AutoCAD® versus Quake® Geometry 49
Figure 4. Extruding a Plane to Form a Solid 49
Figure 5. QERadiant 51
Figure 6. Example of Biker's Unmodified Characteristics 55
Figure 7. Example of Biker's Modified Characteristics 55
Figure 8. Biker's Original Weapon Values 56
Figure 9. Biker's Modified Weapon Values 56
Figure 10. Biker's Unmodified Chat Lines 58
Figure 11. Biker's Modified Chat Lines 58
Figure 12. Biker's Unmodified Alternate Weapon Weights 59
Figure 13. Biker's Modified Alternate Weapon Weights 59
Figure 14. Laser Designator in MilkShape 3D 61
Figure 15. Control File for Laser Designator 62
Figure 16. Laser Designator Model in the Game 63
Figure 17. Picture of Model Tag in MilkShape 3D 64
Figure 18. Biker's Skin Before Editing 66
Figure 19. Biker's Skin After Editing 66
Figure 20. Biker's Original Head Texture 67
Figure 21. Biker's Head Texture With New Head and Ear Added 68
Figure 22. Biker's New Head Texture 68
Figure 23. Biker's New Clothing Texture 69
Figure 24. The New Biker Model in the Game 70
Figure 25. Removing Damage Caused By Railgun 71
Figure 26. Eliminating Railgun Sounds 71
Figure 27. Increasing Railgun Firing Rate 72
Figure 28. Removing Ammunition From Unnecessary Weapons 73
Figure 29. One Life Source Code 74
Figure 30. Modified Top Level Menu 75
Figure 31. Rate Player Menu 75

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1: Summary of Game Evaluations 34

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

3D	Three Dimensional
AI	Artificial Intelligence
BOT	Computer-controlled character (i.e., robot)
BSP	Binary Space Partition
CAD	Computer-Aided Design
COTS	Commercial, Off-the-Shelf (used to describe software)
CTF	Capture the Flag
FPS	First Person Shooter
MMP	Massively Multiplayer
MOD	Game Modification
NDL	Numerical Design Limited
PC	Personal Computer
PvP	Player Versus Player
PVS	Potentially Visible Set
Q3A	Quake III Arena
RAM	Random Access Memory
RPG	Role Playing Game
SKIN	Player's Appearance (stored as a texture)
TD	Team Deathmatch
UT	Unreal Tournament
UI	User Interface
VE	Virtual Environment
VR	Virtual Reality

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENT

The authors would like to acknowledge the financial support of Dr. Casey Wardynski at the Office of Economic & Manpower Analysis for allowing the purchase of equipment and software used in this thesis.

The authors would also like to thank Dr. Michael Capps for his direction and enthusiasm for this thesis. The entire NPSNET research group has been essential in the successful completion of this thesis. Additionally, David Falby was very helpful with his game "evaluation" assistance.

Lastly, we would like to thank our families for their support and understanding: LT DeBrine would like to thank Ami for her love and tolerance during this seemingly endless task, along with his two children for support. LT Morrow would like to thank Suzanne for her love, support and assistance during these last few months.

THIS PAGE INTENTIONALLY LEFT BLANK

I. OVERVIEW

A. THESIS STATEMENT

By re-purposing commercial off-the-shelf (*COTS*) entertainment software, it is possible to produce viable military virtual environment applications that are less expensive yet more engaging than current computer based options.

B. MOTIVATION

Virtual environment and virtual reality applications are achieving widespread use in the military. Military VE applications include theater planning, training, and architectural walkthroughs. The primary benefit of VE applications is that they provide an environment that is safe and can simulate virtually any real-world situation at any time. These applications are currently relatively expensive and inflexible in that a single computer is often dedicated to one type of application, and that very application is often developed specifically for that single task.

1. Military VE Training Applications

There are several types of computer-based VE trainers. At the simplest level are single-user applications where the other members of the team in the trainer are computer-controlled agents. A good example of this type of trainer would be a flight simulator where the other planes in the pilot's formation are computer controlled. The second level brings interactivity between team members. These types of trainers utilize networked applications where each member of the team is presented with its own view of the environment (for example, the same flight simulator, except that the other planes in the formation are controlled by other members of the team). The highest level of computer-based team trainers today are full-scale mockups of an environment where the users may not even realize that they are interacting with a computer at all, such as a Combat Information Center Team Trainer Mockup.

These trainers are very specialized and are often developed for one specific mission or task. As a result, their development can be expensive and time consuming. Usually there is no mechanism to modify the trainer to reflect different changes or desires, other than the "canned" changes that have been programmed into the system. For example, a popular computer-based team trainer is the mock-up of the control room for a nuclear power plant. The computers that control the simulations have been pre-programmed with certain casualties. Introducing a new casualty into the system requires reprogramming the computers. Changing the physical layout of the mock-up to reflect changes to the real-world control room is potentially even more difficult. These types of trainers, while they offer enhanced realism, are usually impossible to modify once they have been constructed.

2. Architectural Walkthroughs

Architectural walkthroughs of virtual environments have been popular applications in the VE/VR field for several years. These applications help users spatially visualize an environment by presenting a three-dimensional (*3D*) representation of the environment. These representations allow the user to navigate around in that environment. The military, along with civilian industry, uses 3D architectural walkthroughs when designing and prototyping buildings, vehicles, and other real-world environments to help determine their habitability. Two of the more famous architectural

VI. ANALYSIS

This chapter explores the benefits of the modification that we have produced. Included is a summary of the entire body of work, along with the benefits of our work and our conclusions and recommendations. The final section contains ideas for future work.

A. SUMMARY OF WORK

Virtual environment and virtual reality applications are achieving widespread use in the military. Unfortunately, these applications are expensive and have a lengthy development time. We set out to determine if a commercial game could be modified to provide at least the same performance, but at a cheaper cost and with faster development time.

After identifying all of our options, we chose a game to be modified. Next, we developed a scenario that would be the goal of our modification: a team trainer. This mock application could later be made into a training device, but our goal was not to actually incorporate training metrics into the application. Rather, we intended to determine if the modifications were possible, and, if possible, what steps were involved. Once our scenario design was completed, we had to design a level. We chose a real-world building converted from CAD data, since realism was important to our application. After our level was complete, we changed certain models in the game. One change was to incorporate a laser designator for the players to use to point out objectives to other players. Also, all unnecessary weapons were removed from the game to enhance realism. The next step in our modification was changing one of the player's skins to resemble civilian clothing. This was also an important element for promoting realism. Once all the model changes and player changes were made, we had to change certain elements of the gameplay, such as giving all players only a single life. The last change to be made was to alter the user interface. We added some menu choices that will be useful when team metrics are incorporated into the scenario.

The final product was a proof-of-concept mock application that achieved our goal. We showed that a commercial game could be modified relatively easily and inexpensively, while providing superior graphics and performance. Also, these modifications can themselves be altered to change the application as requirements shift. Commercial game modifications provide a viable alternative to producing applications from scratch for military use.

B. BENEFITS OF REPURPOSING

1. High Fidelity and Fast Performance

Today's game engines are extremely powerful. They are capable of displaying large architectural datasets with very high fidelity, even real-world buildings. Frame rates are well above thirty frames per second on average hardware platforms, which is more than the twenty-four frames per second necessary for visual fusion (Vince, 1992). Writing a modification for a game allows the developer to leverage this technology.

2. Development Time and Cost

Due to this leveraging of technology, mods can be created in a very short amount of time as compared to development from scratch. "Assault on Herrmann Hall" took two graduate students a total of four months to develop (about 300 man hours). These students were not professional programmers, but they had a basic background in computer programming (professional programmers and artists could undoubtedly create a mod in less time, and with more polished results). The tools to create a mod are readily available, high in quality, easy to use, and often available without cost. These tools are often the same tools that the original game production staff used during the game development. For our mod, the only tools that were not free were Microsoft® Visual Studio® 6.0 for the C++ coding (cost of about $100) and Adobe® PhotoShop® 5.5 (cost of about $600). However, there are excellent freeware substitutes available for these products (gcc and The GIMP). In fact, the Q3A source code is packaged with its own freeware compiler, which we did not use. The only other commercial tool that is commonly used when developing mods (and original games) is 3D Studio Max®. Because we did not modify any animation, we did not require 3D Studio Max®.

With so many tools available for free, the development costs become the major expenses in mod creation. Short development time directly leads to smaller overall salaries for the development staff, keeping the overall cost even lower. Development costs for new games often exceeds one million dollars, but the cost for a basic mod can theoretically be for labor only. Professional mods may not cost much more than a few thousand dollars. Our costs included:

- $6000 for two complete PC workstations
- $100 for Microsoft® Visual Studio® 6.0
- $60 for two licensed copies of Q3A
- $600 for Adobe® PhotoShop® 5.5
- $20 for Milkshape 3D (after a 30-day free trial)

This gave us a total development cost of $6780 (labor costs are not included). This is far less than the cost of developing a VE application from scratch.

C. CONCLUSIONS AND RECOMMENDATIONS

We set out to show that it is possible to modify current games, in a short time, using low cost tools, to produce an application that is useful. Currently, games are often created with the intention of being modified by end users and by professional programmers and artists. However, people with little or no real experience in video game development produce the vast majority of mods. This helps illustrate the fact that simple mods are fairly easy to create. Professional mods are more difficult, but with practice even a novice mod developer can produce a stunning product.

It was easier to create “Assault on Herrmann Hall” as a mod of Q3A than as an original product. Using tools that were inexpensive and easy to use, we completed the mod in approximately four months. In fact, it would have been impossible for us to complete this project as an original program. Without the ability to leverage Q3A’s technology, our scenario would have taken at least a year to develop. Even after that year of design and production, our final result could not have approached the quality of a third-generation best-selling game. The fact that id Software only produces games, and they have professional programmers working on those games, means that our mod can take advantage of their expertise, successes and failures. To try and produce a new scenario without leveraging this technology is an inefficient use of resources.

In the process of developing the mod, we evaluated twenty games based on gameplay criteria. This gave us insight into what capabilities current video games have, and also what games have the capability to be modified by the end user. We also researched the current game engines available, documenting their costs and capabilities. These evaluations provide a mod developer with a place to start when deciding which game to modify for a new application. Our hope is that these evaluations will be stored in a database that can be kept updated for military use. Some kind of central game evaluation repository would be beneficial to those who want to make an informed decision about which game to modify without having to evaluate every game that is currently available.

This thesis has shown that when there is an expressed desire for an application requiring networked interaction between users in real time, serious consideration should be given to modifying current video games to suit that need. Applications that have the best potential for implementation as a game mod are those that require any or all of the following characteristics:

- networking
- extensive use of computer graphics, especially building interiors or barren landscapes
- use of computer-controlled AI characters
- a well-defined world or map that users interact

Another route that may be taken when developing games, or applications similar to games, is to start with a game engine and write an application that uses that engine. This method is considerably more expensive than modifying a current game, but when there are no games similar to the intended application, this may be the only recourse. Choosing this method is still normally faster and less expensive than developing the application from scratch.

As previously stated, modifications allow the developer to leverage the technology currently available in commercial games for a fraction of the cost. In this era of reduced budgets for military simulations and trainers, the modification of commercial games is an attractive alternative to building an application from scratch.

D. FUTURE WORK

While our mock application achieved the goals that were intended, there is more work that can be done to improve understanding of the modification process. In this section we will list some of our suggestions for future work.

1. Implement Training Metrics

One possible area for future work is to modify the source code such that the game actually records leadership data and how well team members follow orders. This type of data would be useful to the military, but would be difficult to implement. No game currently on the market is able to automatically measure team interaction metrics. This also applies to military computer-based trainers.

2. Build a Team Trainer

Our mock application shows that it is possible to make extensive changes to an existing game and create a new application with that game. "Assault on Herrmann Hall" needs additional code changes to be useful as a real team trainer. Teamwork monitoring needs implementing, which fits into the previously mentioned future work of implementing training metrics. Once the trainer is built, a usability study would be an appropriate next step.

3. Build a Theater Planner

We have shown that it is possible to modify a COTS video game to serve as both an architectural walkthrough and as a team trainer. The third implementation of VE/VR software that the military uses is theater planning. It would be very interesting to see if it is possible to modify an existing game to serve as a military theater planner.

4. Build a Game Evaluation Database

As previously mentioned, some kind of central game evaluation repository would be beneficial to those who want to make their own mod. Searching for the optimal game to modify for a certain application would be easier with all evaluations stored in one place and updated as new games are produced.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A.CONVERTER CODE

```
// dxf2map.cpp : Defines the entry point for the console application.
//
// Skip Morrow
// Jeff DeBrine
// 15 Aug 2000
//
// Converts an AutoCAD dxf file to a map file that can
// be read by worldcraft.
//

#include "stdafx.h"
#include <stdlib.h>
#include <iostream>
#include <iomanip.h>
#include <fstream.h>
#include <math.h>
#include <string>
#include <vector>

//using namespace std;

struct coordinate{
    float fX, fY, fZ;
};

// holds four coordinates
struct fourtuple {
    coordinate coord1, coord2, coord3, coord4;
};

// this subtracts two coordinates
coordinate subtract(coordinate c1, coordinate c2)
{
    coordinate tCoord;
    tCoord.fX = c1.fX - c2.fX;
    tCoord.fY = c1.fY - c2.fY;
    tCoord.fZ = c1.fZ - c2.fZ;
    return (tCoord);
}

// calculates the cross product
coordinate cross(coordinate c1, coordinate c2)
{
    coordinate tCoord;
    float fMag;
    tCoord.fX = c1.fY * c2.fZ - c1.fZ * c2.fY;
    tCoord.fY = -(c1.fX * c2.fZ - c1.fZ * c2.fX);
    tCoord.fZ = c1.fX * c2.fY - c2.fX * c1.fY;

    fMag = sqrt(tCoord.fX * tCoord.fX +
                tCoord.fY * tCoord.fY +
                tCoord.fZ * tCoord.fZ);

    tCoord.fX = tCoord.fX / fMag;
    tCoord.fY = tCoord.fY / fMag;
    tCoord.fZ = tCoord.fZ / fMag;
    return (tCoord);
}

// calculates the dot product of two coordinates
float dot(coordinate c1, coordinate c2)
{
    return(c1.fX * c2.fX + c1.fY * c2.fY + c1.fZ * c2.fZ);
}

// Face coords must be written clockwise as seen from the outside of
// the brush.  To ensure that they are written clockwise, choose a
// point from the oposite side of the brush as the viewing point.  This
// point could just as effectively be *inside* the brush, so I know
// that the points should appear counter-clockwise from this point.  To
// find out if three points in 3d space are clockwise or
// counterclockwise, first choose a vertex among the three points.
// Subtract the vertex from the other two points, and you have two
// vectors with their origins at the vertex point. Next subract the
// vertex from the view point, and you have a vectr from the vertex to
// the view point.  Take the cross product of the first two vectors,
// and then the dot product of that cross product and the vector to the
// viewpoint.  If the dot product is negative, the points are clockwise
// as viewed from the outside of the brush.  Write the points to the
// file in the correct order based on the sign (+/-) of the dot product

void writeClockwise(coordinate c1, coordinate c2, coordinate c3,
                                              coordinate vis, ofstream &
ofile)
{
    coordinate v1, v2, vCross, vDot;
    float dProd;
    v1 = subtract(c2, c1);
    v2 = subtract(c3, c1);
    vCross = cross(v1, v2);
    vDot = subtract(vis, c1);
    dProd = dot(vCross, vDot);

    if (dProd > 0) // write the coords in this order
    {
      ofile << "( " << c1.fX << " " << c1.fY << " " << c1.fZ << " ) ( "
                 << c2.fX << " " << c2.fY << " " << c2.fZ << " ) ( "
                 << c3.fX << " " << c3.fY << " "<< c3.fZ << " ) "
                 << "base_wall/concrete 0 0 0 1.0 1.0 0 0 0" << endl;
    }
    else // or this order, depending on the sign of the dot product
    {
     ofile << "( " << c1.fX << " " << c1.fY << " " << c1.fZ << " ) ( "
                   << c3.fX << " " << c3.fY << " " << c3.fZ << " ) ("
                   << c2.fX << " " << c2.fY << " " << c2.fZ << " ) "
                   << "base_wall/concrete 0 0 0 1.0 1.0 0 0 0" << endl;
    }
}

// Takes a plane, extrudes it 2 units along its surface normal
// and makes it a solid by calculating the other five planes
```

```
void makeSolid(fourtuple ft, ofstream & of)
{
    coordinate v1, v2, v;
    fourtuple bf; // backface
    float fExtrude = 2.0; // distance to extrude the plane

    // first get two vectors on the plane
    v1 = subtract(ft.coord1, ft.coord2);
    v2 = subtract(ft.coord1, ft.coord3);

    v = cross(v1, v2); // get the surface normal


    // calculate the coordinates of all the verticies on the
    // plane on the oposite side (four more points in 3d space)
    bf.coord1.fX = ft.coord1.fX + v.fX * fExtrude;
    bf.coord1.fY = ft.coord1.fY + v.fY * fExtrude;
    bf.coord1.fZ = ft.coord1.fZ + v.fZ * fExtrude;

    bf.coord2.fX = ft.coord2.fX + v.fX * fExtrude;
    bf.coord2.fY = ft.coord2.fY + v.fY * fExtrude;
    bf.coord2.fZ = ft.coord2.fZ + v.fZ * fExtrude;

    bf.coord3.fX = ft.coord3.fX + v.fX * fExtrude;
    bf.coord3.fY = ft.coord3.fY + v.fY * fExtrude;
    bf.coord3.fZ = ft.coord3.fZ + v.fZ * fExtrude;

    bf.coord4.fX = ft.coord4.fX + v.fX * fExtrude;
    bf.coord4.fY = ft.coord4.fY + v.fY * fExtrude;
    bf.coord4.fZ = ft.coord4.fZ + v.fZ * fExtrude;


    // write all of the coordinates - must be clockwise!
    of << "{" << endl;
    //front
    writeClockwise(ft.coord1, ft.coord2, ft.coord3, bf.coord1, of);
    //back
    writeClockwise(bf.coord1, bf.coord2, bf.coord3, ft.coord1, of);
    //top
    writeClockwise(bf.coord1, bf.coord2, ft.coord2, bf.coord4, of);
    //bottom
    writeClockwise(ft.coord4, ft.coord3, bf.coord3, ft.coord1, of);
    //left
    writeClockwise(ft.coord2, bf.coord2, bf.coord3, ft.coord1, of);
    //right
    writeClockwise(bf.coord1, ft.coord1, ft.coord4, bf.coord2, of);
    of << "}" << endl;
}


int main(int argc, char* argv[])
{
    ifstream inFile;
    ofstream outFile;

    char cGroupCode[255];
char cValue[255];
int iGroupCode;
float fValue;
float fScaleX = 32.0;
float fScaleY = 32.0;
float fScaleZ = 32.0;
cValue[0] = NULL;
int iCoordCount = 0;
int iFaceCount = 0;
long int iLineNumber = 0;
int iBrushCount = 0;

coordinate tempCoordinate;
fourtuple tempfourtuple;


// open the input file
if (!argv[1])
{
    argv[1] = "inFile.dxf";  // default file name
} // end if


cout << "Opening " << argv[1] << "." << endl;
inFile.open(argv[1], ios::nocreate);

if (!inFile)
{
    cout << "Error!  Could not open " << argv[1] << endl;
} // end if


// open the output file
if (!argv[2])
{
    argv[2] = "outFile.map";  // default file name
} // end if

cout << "Creating " << argv[2] << "." << endl;
outFile.open(argv[2]);

if (!outFile)
{
    cout << "Error!  Could not create " << argv[2] << endl;
} // end if


// write the worldspawn entry at the top
// note that the last outfile is one long line.  Change this line
// as needed to match the correct worldspawn that you need.
outFile << "{" << endl;
outFile << "\"classname\" \"worldspawn\"" << endl;
outFile << "\"wad\" \"\\sierra\\half-life\\valve\\halflife.wad;"
        << "\\sierra\\half-life\\valve\\liquids.wad;"
        << "\\sierra\\half-life\\valve\\xeno.wad;"
        << "\\sierra\\half-life\\valve\\decals.wad\"" << endl;
    // First, find the ENTITIES section (where the geometry is stored)
    // in the dxf file.  There's nothing before this line that I need
    bool bFound = false;
    while (!bFound)
    {
        inFile.getline(cGroupCode, 254);
        iLineNumber++;
        inFile.getline(cValue, 254);
        iLineNumber++;
        iGroupCode = atoi(cGroupCode);

        if ((iGroupCode == 2) && (!strcmp(cValue, "ENTITIES")))
        {
            cout << "Entities Section Found" << endl;
            bFound = true;
        }
    }

    cout << setiosflags(ios::fixed) << setprecision(2);
    outFile << setiosflags(ios::fixed) << setprecision(0);

    // Now, start looking for geometry
    while (inFile.getline(cGroupCode, 254))
    {
        iLineNumber++;
        inFile.getline(cValue, 254);
        iLineNumber++;
        iGroupCode = atoi(cGroupCode);
        fValue = atof(cValue);

        if ((iGroupCode == 0) && ((!strcmp(cValue, "3DFACE")) ||
                                  (!strcmp(cValue, "POLYLINE"))))
        {
//          cout << "3DFACE found" << endl;
            iCoordCount = 0;
        }

        // it is a coordinate for a 3DFACE
        if ((iGroupCode > 9) && (iGroupCode < 40) && (iCoordCount <
                                                                  13))
        {
            iCoordCount++;

            // "x" coords have a group code between 10 and 20
            if (iGroupCode < 20)
            {
                tempCoordinate.fX = fValue * fScaleX;
            }

            // "z" coords have a group code between 30 and 40
            if (iGroupCode > 29)
            {
                tempCoordinate.fZ = fValue * fScaleZ;
            }

            // "y" coords have a group code between 20 and 30
            if ((iGroupCode > 19) && (iGroupCode < 30))
            {
                tempCoordinate.fY = fValue * fScaleY;
            }

            if (iCoordCount == 3)
            {
                tempfourtuple.coord1=tempCoordinate;
            }

            if (iCoordCount == 6)
            {
                tempfourtuple.coord2=tempCoordinate;
            }

            if (iCoordCount == 9)
            {
                tempfourtuple.coord3=tempCoordinate;
            }

            // I now have four coordinates
            if (iCoordCount == 12)
            {
                iCoordCount = 0;
                tempfourtuple.coord4=tempCoordinate;

           // Definitely a hack.  I was getting some strange brushes
           // because sometimes the first two coordinates were the same
           // or I was getting some really high z values.  I chose to
           // just not write those coordinates.  Out of a 20,000
           // polygon model, such as Herrmann Hall, I only lost about
           // 50 polygons this way.  I don't know if it was because of
           // a problem in this code, or if it was an artifact in the
           // dxf file.
/*              if (!((tempfourtuple.coord1.fX ==
                                           tempfourtuple.coord2.fX) &&
                    (tempfourtuple.coord1.fY ==
                                           tempfourtuple.coord2.fY) &&
                    (tempfourtuple.coord1.fZ ==
                                          tempfourtuple.coord2.fZ)) &&
                    (tempfourtuple.coord1.fZ < 200) &&
                    (tempfourtuple.coord2.fZ < 200) &&
                    (tempfourtuple.coord3.fZ < 200) &&
                    (tempfourtuple.coord4.fZ < 200)) */
                if (!((tempfourtuple.coord1.fX ==
                                           tempfourtuple.coord2.fX) &&
                    (tempfourtuple.coord1.fY ==
                                           tempfourtuple.coord2.fY) &&
                    (tempfourtuple.coord1.fZ ==
                                           tempfourtuple.coord2.fZ)))
                {
                    outFile << "// brush " << iBrushCount << "\n";
                    iBrushCount++;
                    makeSolid(tempfourtuple, outFile);
                }
            }
        }
```

```
    }
    cout << "Done reading input file\n";
    outFile << "}" << endl;

    inFile.close();

    return 0;
}
// eof dxf2map
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. Q3A ARTIFICIAL INTELLIGENCE CODE

```
//===========================================================================
// ***MODIFIED CODE***
// Name:                  Biker_c.c
// Function:              Biker, rank 2
// Programmer:            MrElusive (MrElusive@idsoftware.com)
// version:               1
// Tab Size:              4 (real tabs)
//===========================================================================

#include "chars.h"

skill 1
{
	CHARACTERISTIC_NAME                         "Biker"
	CHARACTERISTIC_GENDER                       "male"
	CHARACTERISTIC_ATTACK_SKILL                 0.0
	CHARACTERISTIC_WEAPONWEIGHTS                "bots/biker_w.c"
	CHARACTERISTIC_AIM_SKILL                    0.0
	CHARACTERISTIC_AIM_ACCURACY                 0.0
	CHARACTERISTIC_VIEW_FACTOR                  0.0
	CHARACTERISTIC_VIEW_MAXCHANGE               180
	CHARACTERISTIC_REACTIONTIME                 5.0

	CHARACTERISTIC_CHAT_FILE                    "bots/biker_t.c"
	CHARACTERISTIC_CHAT_NAME                    "biker"
	CHARACTERISTIC_CHAT_CPM                     400
	CHARACTERISTIC_CHAT_INSULT                  0.0
	CHARACTERISTIC_CHAT_MISC                    0.0
	CHARACTERISTIC_CHAT_STARTENDLEVEL           0.5
	CHARACTERISTIC_CHAT_ENTEREXITGAME           0.5
	CHARACTERISTIC_CHAT_KILL                    0.0
	CHARACTERISTIC_CHAT_DEATH                   0.0
	CHARACTERISTIC_CHAT_ENEMYSUICIDE            0.0
	CHARACTERISTIC_CHAT_HITTALKING              0.0
	CHARACTERISTIC_CHAT_HITNODEATH              0.0
	CHARACTERISTIC_CHAT_HITNOKILL               0.0
	CHARACTERISTIC_CHAT_RANDOM                  0.0
	CHARACTERISTIC_CHAT_REPLY                   0.0

	CHARACTERISTIC_CROUCHER                     0.0
	CHARACTERISTIC_JUMPER                       0.0
	CHARACTERISTIC_WEAPONJUMPING                0.0
	CHARACTERISTIC_GRAPPLE_USER                 0.0

	CHARACTERISTIC_ITEMWEIGHTS
	"bots/biker_i.c"
	CHARACTERISTIC_AGGRESSION                   0.0
	CHARACTERISTIC_SELFPRESERVATION             1.0
	CHARACTERISTIC_VENGEFULNESS                 0.0
	CHARACTERISTIC_CAMPER                       0.0

	CHARACTERISTIC_EASY_FRAGGER                 0.0
	CHARACTERISTIC_ALERTNESS                    0.0
}
```

```
skill 4
{
        CHARACTERISTIC_NAME                             "Biker"
        CHARACTERISTIC_GENDER                           "male"
        CHARACTERISTIC_ATTACK_SKILL                     0.0
        CHARACTERISTIC_WEAPONWEIGHTS                    "bots/biker_w.c"
        CHARACTERISTIC_AIM_SKILL                        0.0
        CHARACTERISTIC_AIM_ACCURACY                     0.0
        CHARACTERISTIC_VIEW_FACTOR                      0.0
        CHARACTERISTIC_VIEW_MAXCHANGE                   180
        CHARACTERISTIC_REACTIONTIME                     5.0

        CHARACTERISTIC_CHAT_FILE                        "bots/biker_t.c"
        CHARACTERISTIC_CHAT_NAME                        "biker"
        CHARACTERISTIC_CHAT_CPM                         400
        CHARACTERISTIC_CHAT_INSULT                      0.0
        CHARACTERISTIC_CHAT_MISC                        0.0
        CHARACTERISTIC_CHAT_STARTENDLEVEL               0.5
        CHARACTERISTIC_CHAT_ENTEREXITGAME               0.5
        CHARACTERISTIC_CHAT_KILL                        0.0
        CHARACTERISTIC_CHAT_DEATH                       0.0
        CHARACTERISTIC_CHAT_ENEMYSUICIDE                0.0
        CHARACTERISTIC_CHAT_HITTALKING                  0.0
        CHARACTERISTIC_CHAT_HITNODEATH                  0.0
        CHARACTERISTIC_CHAT_HITNOKILL                   0.0
        CHARACTERISTIC_CHAT_RANDOM                      0.0
        CHARACTERISTIC_CHAT_REPLY                       0.0

        CHARACTERISTIC_CROUCHER                         0.0
        CHARACTERISTIC_JUMPER                           0.0
        CHARACTERISTIC_WEAPONJUMPING                    0.0
        CHARACTERISTIC_GRAPPLE_USER                     0.0

        CHARACTERISTIC_ITEMWEIGHTS                      "bots/biker_i.c"
        CHARACTERISTIC_AGGRESSION                       0.0
        CHARACTERISTIC_SELFPRESERVATION                 1.0
        CHARACTERISTIC_VENGEFULNESS                     0.0
        CHARACTERISTIC_CAMPER                           0.0

        CHARACTERISTIC_EASY_FRAGGER                     0.0
        CHARACTERISTIC_ALERTNESS                        0.0
}

skill 5
{
        CHARACTERISTIC_NAME                             "Biker"
        CHARACTERISTIC_GENDER                           "male"
        CHARACTERISTIC_ATTACK_SKILL                     0.0
        CHARACTERISTIC_WEAPONWEIGHTS                    "bots/biker_w.c"
        CHARACTERISTIC_AIM_SKILL                        0.0
        CHARACTERISTIC_AIM_ACCURACY                     0.0
        CHARACTERISTIC_VIEW_FACTOR                      0.0
        CHARACTERISTIC_VIEW_MAXCHANGE                   240
        CHARACTERISTIC_REACTIONTIME                     5.0

        CHARACTERISTIC_CHAT_FILE                        "bots/biker_t.c"
        CHARACTERISTIC_CHAT_NAME                                "biker"
        CHARACTERISTIC_CHAT_CPM                                 400
        CHARACTERISTIC_CHAT_INSULT                              0.0
        CHARACTERISTIC_CHAT_MISC                                0.0
        CHARACTERISTIC_CHAT_STARTENDLEVEL                       0.5
        CHARACTERISTIC_CHAT_ENTEREXITGAME                       0.5
        CHARACTERISTIC_CHAT_KILL                                0.0
        CHARACTERISTIC_CHAT_DEATH                               0.0
        CHARACTERISTIC_CHAT_ENEMYSUICIDE                        0.0
        CHARACTERISTIC_CHAT_HITTALKING                          0.0
        CHARACTERISTIC_CHAT_HITNODEATH                          0.0
        CHARACTERISTIC_CHAT_HITNOKILL                           0.0
        CHARACTERISTIC_CHAT_RANDOM                              0.0
        CHARACTERISTIC_CHAT_REPLY                               0.0

        CHARACTERISTIC_CROUCHER                                 0.0
        CHARACTERISTIC_JUMPER                                   0.0
        CHARACTERISTIC_WEAPONJUMPING                            0.0
        CHARACTERISTIC_GRAPPLE_USER                             0.0

        CHARACTERISTIC_ITEMWEIGHTS                              "bots/biker_i.c"
        CHARACTERISTIC_AGGRESSION                               0.0
        CHARACTERISTIC_SELFPRESERVATION                         1.0
        CHARACTERISTIC_VENGEFULNESS                             0.0
        CHARACTERISTIC_CAMPER                                   0.0

        CHARACTERISTIC_EASY_FRAGGER                             0.0
        CHARACTERISTIC_ALERTNESS                                0.0
}
```

```
//=========================================================================
// ***MODIFIED CODE***
// Name:                visor_i.c
// Function:
// Programmer:          Mr Elusive (MrElusive@idsoftware.com)
// Last update:         1999-09-08
// Tab Size:            4 (real tabs)
//=========================================================================

#include "inv.h"

//initial health/armor states
#define FS_HEALTH                                         1
#define FS_ARMOR                                          1

//initial weapon weights
#define W_SHOTGUN                                         1
#define W_MACHINEGUN                                      1
#define W_GRENADELAUNCHER                                 1
#define W_ROCKETLAUNCHER                                  1
#define W_RAILGUN                                         1
#define W_BFG10K                                          1
#define W_LIGHTNING                                       1
#define W_PLASMAGUN                                       1

//the bot has the weapons, so the weights change a little bit
#define GWW_SHOTGUN                                       1
#define GWW_MACHINEGUN                                    1
#define GWW_GRENADELAUNCHER                               1
#define GWW_ROCKETLAUNCHER                                1
#define GWW_RAILGUN                                       1
#define GWW_BFG10K                                        1
#define GWW_LIGHTNING                                     1
#define GWW_PLASMAGUN                                     1

//initial powerup weights
#define W_TELEPORTER                                      1
#define W_MEDKIT                                          1
#define W_QUAD                                            1
#define W_ENVIRO                                          1
#define W_HASTE                                           1
#define W_INVISIBILITY                                    1
#define W_REGEN                                           1
#define W_FLIGHT                                          1

//flag weight
#define FLAG_WEIGHT                                       1

//
#include "fw_items.c"
```

```
//======================================================================
// ***MODIFIED CODE***
// Name:                    Biker_c.c
// Function:                chat lines for Biker
// Programmer:              MrElusive (MrElusive@idsoftware.com)
// Author:                  The Seven Swords & R.A. Salvatore
// Editor:                  Paul Jaquays
// Last update:             Oct. 10, 1999
// Tab Size:                3 (real tabs)
//======================================================================

//example initial chats
chat "biker"
{
      //the teamplay.h file is included for all kinds of teamplay chats
      #include "teamplay.h"

      //=========================================================
      //=========================================================

      type "game_enter" //initiated when the bot enters the game
      {
            HELLO5;
            "I am a great hostage.";
/*          "Make this a good day, cuz it'll be your last.";
            "Everyone out of the pool.";
            "You just keep your hands off my hawg, ", 1, ".
Understand?";
            1, ", I'm gonna kick your sorry butt into the next
county.";
            "I squash cockroaches bigger and scarier than you, ", 1,
".";
            "This better not make me late for lunch.";  */
            // 0 = bot name
      } //end type

      type "game_exit" //initiated when the bot exits the game
      {
            "I have some boredom to catch up on.";
            "Screw this.";
            GOODBYE1;
            // 0 = bot name
      } //end type

      type "level_start" //initiated when a new level starts
      {
            LEVEL_START1;
            "I'm makin' ", 4, " my playground. Get off the swings, ",
fighter, "s.";
            "Where'd I park my hawg?";
            // 0 = bot name
            // 1 = randomly chosen player
            // 4 = Level's title
            } //end type

      type "level_end" //initiated when a level ends and the bot is not
first and not last in the rankings
```

```
      {
            "Waste of time.";
            LEVEL_END0;
            2, " cheats and ", 3, " sucks. Now that we know, let's go
again.";
            // 0 = bot name
            // 2 = opponent in first place
            // 3 = opponent in last place
      } //end type

      type "level_end_victory" //initiated when a level ends and the
bot is first in the rankings
      {
            "Yeehaa! Kick some doggie butt!";
            "Did that hurt ya, ", 3, "?  Bet it did!";
            LEVEL_END_VICTORY3;
            // 0 = bot name
            // 3 = opponent in last place
      } //end type

      type "level_end_lose" //initiated when a level ends and the bot
is last in the rankings
      {
            "Aw, you can kiss my bike wheel ", 2, " .... but lemme get
it spinnin' real fast first.";
            "That's it! I'm putting me a rifle rack on my hawg.";

            LEVEL_END_LOSE1;
            // 0 = bot name
      } //end type

      //=========================================================
      //=========================================================

      type "hit_talking" //bot is hit while chat balloon is visible;
lecture attacker on poor sportsmanship
      {
            //"You just became road-kill, ", fighter, ".";
            0, ", you just became a kick-stand.";
            "Nice ~one, ", 0, ". But now it's my turn.";
            //0 = shooter
      } //end type

      type "hit_nodeath" //bot is hit by an opponent's weapon attack;
either praise or insult
      {
            "C'mon, dawg, I still gots ~one good eye!";
            "Don't dirty the leathers, ", 0,"!";
            "You cow-poker! I'll show you what's what!";
            "OK, you gave it your best and now it's my turn.";
            "I'm gonna hurt you, ", 0, ". I'm gonna hurt you bad.";
            DEATH_INSULT2;
            //0 = shooter
      } //end type

      type "hit_nokill" //bot hits an opponent but does not kill it
      {
```

```
		TAUNT1;
		"Yer as slow as ya are ugly.";
		TAUNT8;
		"You look mighty pretty in blood red.";
		"Lousy, cheap ammo.";
		"Sheee-oooot! I gotta get me a bigger gun than this ~one.";
		//0 = opponent
	} //end type

	//=======================================================
	//=======================================================

	type "death_telefrag" //initiated when the bot is killed by a
telefrag
	{
		"Them trucks never give a hawg-rider no respect.";
		"That was some serious road rash.";
		"Anybody get the license plate on that elephant?";
		DEATH_TELEFRAGGED0;
		// 0 = enemy name
	} //end type

	type "death_cratered" //initiated when the bot is killed by
falling damage
	{
		"Them Kneivel boys ain't got nothin' on me.";
		"May be somethin' to them helmet laws ...";
		"Dang! Colorado Rocky Mountain high!";
		// 0 = enemy name
	} //end type

	type "death_lava" //initiated when the bot dies in lava
	{
		"Dang! That's hotter than a tailpipe runnin' jet fuel.";
		"Hoooooweeee! Too much lighter fluid on the grill there.";
		DEATH_LAVA1;
		// 0 = enemy name
	} //end type

	type "death_slime" //initiated when the bot dies in slime
	{
		"Any biscuits with this gravy?";
		"This is like showerin' in degreaser.";
		"Hey Lucy, this stuff's better'n yer maw's cookin'!";
		// 0 = enemy name
	} //end type

	type "death_drown" //initiated when the bot drowns
	{
		"Leather's heavy when it's wet. Dang.";
		"Great, now I smells like a wet dog.";
		"Stupid place for a swimmmin' pool.";
		// 0 = enemy name
	} //end type

	type "death_suicide" //initiated when bot blows self up with a
weapon or craters
```

```
    {
        "You so much as giggle, ", 0, ", and I break yer face.";
        "Did that truck run over my bike too?";
        DEATH_SUICIDE0;
        DEATH_SUICIDE5;
        // 0 = enemy name
    } //end type

    type "death_gauntlet" //initiated when the bot is killed by a
    gauntlet attack
    {
        "Get yer slimy paws off o' me, ", 0, "!";
        "Last guy that touched me can only count to ~five now.";
        "Ain't nobody touches me like that.";
        DEATH_GAUNTLET2;
        // 0 = enemy name
    } //end type

    type "death_rail" //initiated when the bot is killed by a rail
gun shot
    {
        DEATH_RAIL2;
        "This ain't no campground, ", 0, ".";
        "Let's settle this outside, face to face.";
        // 0 = enemy name
    } //end type

    type "death_bfg" //initiated when the bot died by a BFG
    {
        "And the friggin' rockets red ... uhhh ... green glare!";
        DEATH_BFG2;
        "Ha, ya missed, ", 0, "! 'Almost' only counts in hand
grenades and ... oh, puke ...";
        "Stampede!";
        // 0 = enemy name
    } //end type

    type "death_insult" //insult initiated when the bot died
    {
        "I take my coffee stronger'n you, ", 0, ".";
        "You just stepped on the highway to hell, ", fighter, ".";
        "I can't believe a ", fighter, " like ", 0, " fragged me!";
        DEATH_INSULT0;
        DEATH_INSULT4;
        DEATH_INSULT5;
        // 0 = enemy name
    } //end type

    type "death_praise" //praise initiated when the bot died
    {
        "Yeah, big hairy deal, ", 0, ". Ya killed me. So what?";
        "Well, ain't ", 0, " a ~hero?  Come and get your special
prize.";
        D_PRAISE2;
        0, " killed me to death! Dang!";
        // 0 = enemy name
    } //end type
```

```
        //=====================================================
        //=====================================================

        type "kill_rail" //initiated when the bot kills someone with rail
gun
        {
                "Turn into the wind, ", fighter, ", I wanna hear ya
whistle.";
                KILL_RAIL1;
                KILL_RAIL0;
                // 0 = enemy name
        } //end type

        type "kill_gauntlet" //initiated when the bot kills someone with
gauntlet
        {
                "Up close and impersonal.";
                "Like swattin’ flies offa roadkill.";
                "That’s the kind of action I like most.";
                "Yeah! Gimme some more of that!";
                KILL_GAUNTLET0;
                // 0 = enemy name
        } //end type

        type "kill_telefrag" //initiated when the bot telefragged someone
        {
                "Here’s the beef!";
                "Get off the Road!";
                "Popped you like a Prairie Oyster!";
                TELEFRAGGED0;
                // 0 = enemy name
        } //end type

        type "kill_suicide" //initiated when the player kills self
        {
                "Hey moron, AIM, then fire!";
                "Must tickle, cuz I’m laughin’";
                // 0 = enemy name
        } //end type

        type "kill_insult" //Insult initiated when the bot killed someone
        {
                "I’m gonna bury ", 0, " face down so I got a place to park
my bike.";
                "Ahhhh, you ain’t worth the trouble to run you over.";
                "Biker 1, Roadkill 0.";
                KILL_INSULT0;
                KILL_INSULT4;
                KILL_INSULT8;
                // 0 = enemy name
        } //end type

        type "kill_praise" //praise initiated when the bot killed someone
        {
                KILL_INSULT36;
                "You’ll make good maggot food, ", 0, ". That’s something.";
```

```
            "You was brain-dead anyway, ", 0, ".";

            // 0 = enemy name
      } //end type

      //=======================================================
      //=======================================================

      type "random_insult" //insult initiated randomly (just when the
bot feels like it)
      {
            TAUNT7;
            MISC6;
            "All the sex appeal of open heart surgery!";
            "Yer like sand on the road, ", 0, ".I just blow it away.";
            "Your mamma’s gonna see you on a milk carton, ", 0, ".";
            "You stay away from my dawg, ", 0, ".";
            // 0 = name of randomly chosen player
            // 1 = bot name
      } //end type

      type "random_misc" //miscellanous chats initiated randomly
      {
            GUYTALK2;
            "I eat my road kill.";
            "*[Bu-u-u-u-u-u-urp]* ";
            "Anybody got a ~six-pack on ’em?";
            "Okay, any o’ you pretty ladies want a ride on my hawg?";
            one_liners;
            "Don’t go tellin’ Lucy I was here ... she’ll whip my
butt.";
            femalebot, " better git her ass back in the kitchen and
finish making me that chicken pot pie!";
            MISC0;
            MISC8;
            MISC12;
            // 0 = name of randomly chosen player
            // 1 = bot name
      } //end type
} //end biker chat
```

```
//======================================================================
// ***MODIFIED CODE***
// Name:                visor_w.c
// Function:
// Programmer:
// Last update:
// Tab Size:            4 (real tabs)
//======================================================================

#include "inv.h"

#define W_GAUNTLET                                          1
#define W_SHOTGUN                                           1
#define W_MACHINEGUN                                        1
#define W_GRENADELAUNCHER                                   1
#define W_ROCKETLAUNCHER                                    1
#define W_RAILGUN                                           1
#define W_BFG10K                                            1
#define W_LIGHTNING                                         1
#define W_PLASMAGUN                                         1
#define W_GRAPPLE                                           1

//
#include "fw_weap.c"
```

```
//===========================================================================
// ***ORIGINAL CODE***
// Name:                Biker_c.c
// Function:            Biker, rank 2
// Programmer:          MrElusive (MrElusive@idsoftware.com)
// version:             1
// Tab Size:            4 (real tabs)
//===========================================================================

#include "chars.h"

skill 1
{
	CHARACTERISTIC_NAME
	"Biker"
	CHARACTERISTIC_GENDER                         "male"
	CHARACTERISTIC_ATTACK_SKILL                   0.4
	CHARACTERISTIC_WEAPONWEIGHTS                  "bots/biker_w.c"
	CHARACTERISTIC_AIM_SKILL                      0.5
	CHARACTERISTIC_AIM_ACCURACY                   0.5
	CHARACTERISTIC_VIEW_FACTOR                    0.5
	CHARACTERISTIC_VIEW_MAXCHANGE                 180
	CHARACTERISTIC_REACTIONTIME                   2.5

	CHARACTERISTIC_CHAT_FILE                      "bots/biker_t.c"
	CHARACTERISTIC_CHAT_NAME                      "biker"
	CHARACTERISTIC_CHAT_CPM                       400
	CHARACTERISTIC_CHAT_INSULT                    0.95
	CHARACTERISTIC_CHAT_MISC                      0.5
	CHARACTERISTIC_CHAT_STARTENDLEVEL             0.5
	CHARACTERISTIC_CHAT_ENTEREXITGAME             0.5
	CHARACTERISTIC_CHAT_KILL                      0.5
	CHARACTERISTIC_CHAT_DEATH                     0.5
	CHARACTERISTIC_CHAT_ENEMYSUICIDE              0.5
	CHARACTERISTIC_CHAT_HITTALKING                0.15
	CHARACTERISTIC_CHAT_HITNODEATH                0.5
	CHARACTERISTIC_CHAT_HITNOKILL                 0.5
	CHARACTERISTIC_CHAT_RANDOM                    0.5
	CHARACTERISTIC_CHAT_REPLY                     0.4

	CHARACTERISTIC_CROUCHER                       0.25
	CHARACTERISTIC_JUMPER                         0.25
	CHARACTERISTIC_WEAPONJUMPING                  0.5
	CHARACTERISTIC_GRAPPLE_USER                   0.5

	CHARACTERISTIC_ITEMWEIGHTS                    "bots/biker_i.c"
	CHARACTERISTIC_AGGRESSION                     0.75
	CHARACTERISTIC_SELFPRESERVATION               0.5
	CHARACTERISTIC_VENGEFULNESS                   0.5
	CHARACTERISTIC_CAMPER                         0.5

	CHARACTERISTIC_EASY_FRAGGER                   0.5
	CHARACTERISTIC_ALERTNESS                      0.5
}
```

```
skill 4
{
	CHARACTERISTIC_NAME                         "Biker"
	CHARACTERISTIC_GENDER                       "male"
	CHARACTERISTIC_ATTACK_SKILL                 0.5
	CHARACTERISTIC_WEAPONWEIGHTS                "bots/biker_w.c"
	CHARACTERISTIC_AIM_SKILL                    0.5
	CHARACTERISTIC_AIM_ACCURACY                 0.5
	CHARACTERISTIC_VIEW_FACTOR                  0.5
	CHARACTERISTIC_VIEW_MAXCHANGE               180
	CHARACTERISTIC_REACTIONTIME                 2.5

	CHARACTERISTIC_CHAT_FILE                    "bots/biker_t.c"
	CHARACTERISTIC_CHAT_NAME                    "biker"
	CHARACTERISTIC_CHAT_CPM                     400
	CHARACTERISTIC_CHAT_INSULT                  0.95
	CHARACTERISTIC_CHAT_MISC                    0.5
	CHARACTERISTIC_CHAT_STARTENDLEVEL           0.5
	CHARACTERISTIC_CHAT_ENTEREXITGAME           0.5
	CHARACTERISTIC_CHAT_KILL                    0.5
	CHARACTERISTIC_CHAT_DEATH                   0.5
	CHARACTERISTIC_CHAT_ENEMYSUICIDE            0.5
	CHARACTERISTIC_CHAT_HITTALKING              0.15
	CHARACTERISTIC_CHAT_HITNODEATH              0.5
	CHARACTERISTIC_CHAT_HITNOKILL               0.5
	CHARACTERISTIC_CHAT_RANDOM                  0.5
	CHARACTERISTIC_CHAT_REPLY                   0.175

	CHARACTERISTIC_CROUCHER                     0.5
	CHARACTERISTIC_JUMPER                       0.5
	CHARACTERISTIC_WEAPONJUMPING                0.5
	CHARACTERISTIC_GRAPPLE_USER                 0.5

	CHARACTERISTIC_ITEMWEIGHTS                  "bots/biker_i.c"
	CHARACTERISTIC_AGGRESSION                   0.5
	CHARACTERISTIC_SELFPRESERVATION             0.5
	CHARACTERISTIC_VENGEFULNESS                 0.5
	CHARACTERISTIC_CAMPER                       0.5

	CHARACTERISTIC_EASY_FRAGGER                 0.5
	CHARACTERISTIC_ALERTNESS                    0.5
}

skill 5
{
	CHARACTERISTIC_NAME                         "Biker"
	CHARACTERISTIC_GENDER                       "male"
	CHARACTERISTIC_ATTACK_SKILL                 0.95
	CHARACTERISTIC_WEAPONWEIGHTS                "bots/biker_w.c"
	CHARACTERISTIC_AIM_SKILL                    1.0
	CHARACTERISTIC_AIM_ACCURACY                 1.0
	CHARACTERISTIC_VIEW_FACTOR                  1.0
	CHARACTERISTIC_VIEW_MAXCHANGE               240
	CHARACTERISTIC_REACTIONTIME                 0.15

	CHARACTERISTIC_CHAT_FILE                    "bots/biker_t.c"
	CHARACTERISTIC_CHAT_NAME                    "biker"
```

```
        CHARACTERISTIC_CHAT_CPM                                 400
        CHARACTERISTIC_CHAT_INSULT                              0.95
        CHARACTERISTIC_CHAT_MISC                                0.5
        CHARACTERISTIC_CHAT_STARTENDLEVEL                       0.5
        CHARACTERISTIC_CHAT_ENTEREXITGAME                       0.5
        CHARACTERISTIC_CHAT_KILL                                0.5
        CHARACTERISTIC_CHAT_DEATH                               0.5
        CHARACTERISTIC_CHAT_ENEMYSUICIDE                        0.5
        CHARACTERISTIC_CHAT_HITTALKING                          0.15
        CHARACTERISTIC_CHAT_HITNODEATH                          0.5
        CHARACTERISTIC_CHAT_HITNOKILL                           0.5
        CHARACTERISTIC_CHAT_RANDOM                              0.5
        CHARACTERISTIC_CHAT_REPLY                               0.1

        CHARACTERISTIC_CROUCHER                                 0.0
        CHARACTERISTIC_JUMPER                                   1.0
        CHARACTERISTIC_WEAPONJUMPING                            0.5
        CHARACTERISTIC_GRAPPLE_USER                             0.5

        CHARACTERISTIC_ITEMWEIGHTS                              "bots/biker_i.c"
        CHARACTERISTIC_AGGRESSION                               0.5
        CHARACTERISTIC_SELFPRESERVATION                         0.5
        CHARACTERISTIC_VENGEFULNESS                             0.5
        CHARACTERISTIC_CAMPER                                   0.5

        CHARACTERISTIC_EASY_FRAGGER                             0.5
        CHARACTERISTIC_ALERTNESS                                0.5
}
```

```
//=======================================================================
// ***ORIGINAL CODE***
// Name:                visor_i.c
// Function:
// Programmer:          Mr Elusive (MrElusive@idsoftware.com)
// Last update:         1999-09-08
// Tab Size:            4 (real tabs)
//=======================================================================

#include "inv.h"

//initial health/armor states
#define FS_HEALTH                                           1
#define FS_ARMOR                                            1

//initial weapon weights
#define W_SHOTGUN                                           50
#define W_MACHINEGUN                                        40
#define W_GRENADELAUNCHER                                   40
#define W_ROCKETLAUNCHER                                    120
#define W_RAILGUN                                           85
#define W_BFG10K                                            30
#define W_LIGHTNING                                         50
#define W_PLASMAGUN                                         500

//the bot has the weapons, so the weights change a little bit
#define GWW_SHOTGUN                                         35
#define GWW_MACHINEGUN                                      50
#define GWW_GRENADELAUNCHER                                 30
#define GWW_ROCKETLAUNCHER                                  90
#define GWW_RAILGUN                                         25
#define GWW_BFG10K                                          41
#define GWW_LIGHTNING                                       40
#define GWW_PLASMAGUN                                       40

//initial powerup weights
#define W_TELEPORTER                                        40
#define W_MEDKIT                                            40
#define W_QUAD                                              40
#define W_ENVIRO                                            40
#define W_HASTE                                             40
#define W_INVISIBILITY                                      40
#define W_REGEN                                             40
#define W_FLIGHT                                            40

//flag weight
#define FLAG_WEIGHT                                         50

//
#include "fw_items.c"
```

```
//======================================================================
//
// Name:                    Biker_c.c
// Function:                chat lines for Biker
// Programmer:              MrElusive (MrElusive@idsoftware.com)
// Author:                  The Seven Swords & R.A. Salvatore
// Editor:                  Paul Jaquays
// Last update:             Oct. 10, 1999
// Tab Size:                3 (real tabs)
//======================================================================

//example initial chats
chat "biker"
{
      //the teamplay.h file is included for all kinds of teamplay chats
      #include "teamplay.h"

      //=======================================================
      //=======================================================

      type "game_enter" //initiated when the bot enters the game
      {
            HELLO5;
            "Make this a good day, cuz it'll be your last.";
            "Everyone out of the pool.";
            "You just keep your hands off my hawg, ", 1, ".
Understand?";
            1, ", I'm gonna kick your sorry butt into the next
county.";
            "I squash cockroaches bigger and scarier than you, ", 1,
".";
            "This better not make me late for lunch.";
            // 0 = bot name
      } //end type

      type "game_exit" //initiated when the bot exits the game
      {
            "I have some boredom to catch up on.";
            "Screw this.";
            GOODBYE1;
            // 0 = bot name
      } //end type

      type "level_start" //initiated when a new level starts
      {
            LEVEL_START1;
            "I'm makin' ", 4, " my playground. Get off the swings, ",
fighter, "s.";
            "Where'd I park my hawg?";
            // 0 = bot name
            // 1 = randomly chosen player
            // 4 = Level's title
            } //end type

      type "level_end" //initiated when a level ends and the bot is not
first and not last in the rankings
      {
            "Waste of time.";
            LEVEL_END0;
            2, " cheats and ", 3, " sucks. Now that we know, let's go
again.";
            // 0 = bot name
            // 2 = opponent in first place
            // 3 = opponent in last place
      } //end type

      type "level_end_victory" //initiated when a level ends and the
bot is first in the rankings
      {
            "Yeehaa! Kick some doggie butt!";
            "Did that hurt ya, ", 3, "?  Bet it did!";
            LEVEL_END_VICTORY3;
            // 0 = bot name
            // 3 = opponent in last place
      } //end type

      type "level_end_lose" //initiated when a level ends and the bot
is last in the rankings
      {
            "Aw, you can kiss my bike wheel ", 2, " .... but lemme get
it spinnin' real fast first.";
            "That's it! I'm putting me a rifle rack on my hawg.";

            LEVEL_END_LOSE1;
            // 0 = bot name
      } //end type

      //=========================================================
      //=========================================================

      type "hit_talking" //bot is hit while chat balloon is visible;
lecture attacker on poor sportsmanship
      {
            //"You just became road-kill, ", fighter, ".";
            0, ", you just became a kick-stand.";
            "Nice ~one, ", 0, ". But now it's my turn.";
            //0 = shooter
      } //end type

      type "hit_nodeath" //bot is hit by an opponent's weapon attack;
either praise or insult
      {
            "C'mon, dawg, I still gots ~one good eye!";
            "Don't dirty the leathers, ", 0,"!";
            "You cow-poker! I'll show you what's what!";
            "OK, you gave it your best and now it's my turn.";
            "I'm gonna hurt you, ", 0, ". I'm gonna hurt you bad.";
            DEATH_INSULT2;
            //0 = shooter
      } //end type

      type "hit_nokill" //bot hits an opponent but does not kill it
      {
            TAUNT1;
            "Yer as slow as ya are ugly.";
            TAUNT8;
            "You look mighty pretty in blood red.";
            "Lousy, cheap ammo.";
            "Sheee-oooot! I gotta get me a bigger gun than this ~one.";
            //0 = opponent
      } //end type

      //=========================================================
      //=========================================================

      type "death_telefrag" //initiated when the bot is killed by a
telefrag
      {
            "Them trucks never give a hawg-rider no respect.";
            "That was some serious road rash.";
            "Anybody get the license plate on that elephant?";
            DEATH_TELEFRAGGED0;
            // 0 = enemy name
      } //end type

      type "death_cratered" //initiated when the bot is killed by
falling damage
      {
            "Them Kneivel boys ain’t got nothin’ on me.";
            "May be somethin’ to them helmet laws ...";
            "Dang! Colorado Rocky Mountain high!";
            // 0 = enemy name
      } //end type

      type "death_lava" //initiated when the bot dies in lava
      {
            "Dang! That’s hotter than a tailpipe runnin’ jet fuel.";
            "Hoooooweeee! Too much lighter fluid on the grill there.";
            DEATH_LAVA1;
            // 0 = enemy name
      } //end type

      type "death_slime" //initiated when the bot dies in slime
      {
            "Any biscuits with this gravy?";
            "This is like showerin’ in degreaser.";
            "Hey Lucy, this stuff’s better’n yer maw’s cookin’!";
            // 0 = enemy name
      } //end type

      type "death_drown" //initiated when the bot drowns
      {
            "Leather’s heavy when it’s wet. Dang.";
            "Great, now I smells like a wet dog.";
            "Stupid place for a swimmmin’ pool.";
            // 0 = enemy name
      } //end type

      type "death_suicide" //initiated when bot blows self up with a
weapon or craters
      {
            "You so much as giggle, ", 0, ", and I break yer face.";
            "Did that truck run over my bike too?";
            DEATH_SUICIDE0;
            DEATH_SUICIDE5;
            // 0 = enemy name
      } //end type

      type "death_gauntlet" //initiated when the bot is killed by a
gauntlet attack
      {
            "Get yer slimy paws off o' me, ", 0, "!";
            "Last guy that touched me can only count to ~five now.";
            "Ain't nobody touches me like that.";
            DEATH_GAUNTLET2;
            // 0 = enemy name
      } //end type

      type "death_rail" //initiated when the bot is killed by a rail
gun shot
      {
            DEATH_RAIL2;
            "This ain't no campground, ", 0, ".";
            "Let's settle this outside, face to face.";
            // 0 = enemy name
      } //end type

      type "death_bfg" //initiated when the bot died by a BFG
      {
            "And the friggin' rockets red ... uhhh ... green glare!";
            DEATH_BFG2;
            "Ha, ya missed, ", 0, "! 'Almost' only counts in hand
grenades and ... oh, puke ...";
            "Stampede!";
            // 0 = enemy name
      } //end type

      type "death_insult" //insult initiated when the bot died
      {
            "I take my coffee stronger'n you, ", 0, ".";
            "You just stepped on the highway to hell, ", fighter, ".";
            "I can't believe a ", fighter, " like ", 0, " fragged me!";
            DEATH_INSULT0;
            DEATH_INSULT4;
            DEATH_INSULT5;
            // 0 = enemy name
      } //end type

      type "death_praise" //praise initiated when the bot died
      {
            "Yeah, big hairy deal, ", 0, ". Ya killed me. So what?";
            "Well, ain't ", 0, " a ~hero?  Come and get your special
prize.";
            D_PRAISE2;
            0, " killed me to death! Dang!";
            // 0 = enemy name
      } //end type
```

```
        //========================================================
        //========================================================

        type "kill_rail" //initiated when the bot kills someone with rail gun
        {
                "Turn into the wind, ", fighter, ", I wanna hear ya whistle.";
                KILL_RAIL1;
                KILL_RAIL0;
                // 0 = enemy name
        } //end type

        type "kill_gauntlet" //initiated when the bot kills someone with gauntlet
        {
                "Up close and impersonal.";
                "Like swattin' flies offa roadkill.";
                "That's the kind of action I like most.";
                "Yeah! Gimme some more of that!";
                KILL_GAUNTLET0;
                // 0 = enemy name
        } //end type

        type "kill_telefrag" //initiated when the bot telefragged someone
        {
                "Here's the beef!";
                "Get off the Road!";
                "Popped you like a Prairie Oyster!";
                TELEFRAGGED0;
                // 0 = enemy name
        } //end type

        type "kill_suicide" //initiated when the player kills self
        {
                "Hey moron, AIM, then fire!";
                "Must tickle, cuz I'm laughin'";
                // 0 = enemy name
        } //end type

        type "kill_insult" //Insult initiated when the bot killed someone
        {
                "I'm gonna bury ", 0, " face down so I got a place to park my bike.";
                "Ahhhh, you ain't worth the trouble to run you over.";
                "Biker 1, Roadkill 0.";
                KILL_INSULT0;
                KILL_INSULT4;
                KILL_INSULT8;
                // 0 = enemy name
        } //end type

        type "kill_praise" //praise initiated when the bot killed someone
        {
                KILL_INSULT36;
                "You'll make good maggot food, ", 0, ". That's something.";
                "You was brain-dead anyway, ", 0, ".";
```

```
            // 0 = enemy name
      } //end type

      //======================================================
      //======================================================

      type "random_insult" //insult initiated randomly (just when the
bot feels like it)
      {
            TAUNT7;
            MISC6;
            "All the sex appeal of open heart surgery!";
            "Yer like sand on the road, ", 0, ".I just blow it away.";
            "Your mamma's gonna see you on a milk carton, ", 0, ".";
            "You stay away from my dawg, ", 0, ".";
            // 0 = name of randomly chosen player
            // 1 = bot name
      } //end type

      type "random_misc" //miscellanous chats initiated randomly
      {
            GUYTALK2;
            "I eat my road kill.";
            "*[Bu-u-u-u-u-urp]* ";
            "Anybody got a ~six-pack on 'em?";
            "Okay, any o' you pretty ladies want a ride on my hawg?";
            one_liners;
            "Don't go tellin' Lucy I was here ... she'll whip my
butt.";
            femalebot, " better git her ass back in the kitchen and
finish making me that chicken pot pie!";
            MISC0;
            MISC8;
            MISC12;
            // 0 = name of randomly chosen player
            // 1 = bot name
      } //end type
} //end biker chat
```

```
//======================================================================
//
// Name:                visor_w.c
// Function:
// Programmer:
// Last update:
// Tab Size:            4 (real tabs)
//======================================================================

#include "inv.h"

#define W_GAUNTLET                                          10
#define W_SHOTGUN                                           30
#define W_MACHINEGUN                                        20
#define W_GRENADELAUNCHER                                   40
#define W_ROCKETLAUNCHER                                    100
#define W_RAILGUN                                           70
#define W_BFG10K                                            95
#define W_LIGHTNING                                         80
#define W_PLASMAGUN                                         275
#define W_GRAPPLE                                           15

//
#include "fw_weap.c"
```

APPENDIX C. Q3A MENU CODE

```
//
/*
============================================================
RATE PLAYER MENU

Skip Morrow
Jeff DeBrine
July 2000

Other Q3A menu files were used as a template
Contains original code for new menu choices
============================================================
*/

#include "ui_local.h"

#define ART_FRAME        "menu/art/addbotframe"
#define ART_BACK0        "menu/art/back_0"
#define ART_BACK1        "menu/art/back_1"

#define RATE_PLAYER_MENU_VERTICAL_SPACING   28

#define ID_EXCELLENT      10
#define ID_AVERAGE        11
#define ID_BELOW_AVERAGE  12
#define ID_POOR           13

typedef struct {
    menuframework_s menu;

    menutext_s      banner;
    menutext_s      excellent;
    menutext_s      average;
    menutext_s      belowAverage;
    menutext_s      poor;
    menubitmap_s    frame;
    menulist_s      list;
    menubitmap_s    back;

    int             gametype;
    int             numBots;
    int             selectedBot;
    char            *bots[9];
    char            botNames[9][16];
} ratePlayerMenuInfo_t;

static ratePlayerMenuInfo_t ratePlayerMenuInfo;

#define NUM_RATINGS     3
static const char *ratings[] = {
    "Excellent",
    "Average",
    "Below Average",
    "Poor",
    NULL
};

/*
===============
UI_RatePlayerMenu_BackEvent
===============
*/
static void UI_RatePlayerMenu_BackEvent( void *ptr, int event ) {
    if( event != QM_ACTIVATED ) {
        return;
    }
    UI_PopMenu();
}

/*
===============
UI_RatePlayerMenu_SetList
===============
*/
static void UI_RatePlayerMenu_SetList( int id ) {
    ratePlayerMenuInfo.list.generic.id = id;
    ratePlayerMenuInfo.list.numitems = NUM_RATINGS;
    ratePlayerMenuInfo.list.itemnames = ratings;
    ratePlayerMenuInfo.list.generic.bottom =
    ratePlayerMenuInfo.list.generic.top +
    ratePlayerMenuInfo.list.numitems * PROP_HEIGHT;
}

/*
=================
RatePlayer_Event
=================
*/
void RatePlayer_Event( void *ptr, int notification ) {
    if( notification != QM_ACTIVATED ) {
        return;
    }

    switch( ((menucommon_s*)ptr)->id ) {
    case ID_EXCELLENT:
        UI_PopMenu();
        break;

    case ID_AVERAGE:
        UI_PopMenu();
        break;

    case ID_BELOW_AVERAGE:
        UI_PopMenu();
        break;

    case ID_POOR:
        UI_PopMenu();
        break;
    }
}
```

```
/*
=================
UI_RatePlayerMenu_Key
=================
*/
sfxHandle_t UI_RatePlayerMenu_Key( int key ) {
    menulist_s  *l;
    int x;
    int y;
    int index;

    l = (menulist_s *)Menu_ItemAtCursor( &ratePlayerMenuInfo.menu );
    if( l != &ratePlayerMenuInfo.list ) {
        return Menu_DefaultKey( &ratePlayerMenuInfo.menu, key );
    }

    switch( key ) {
        case K_MOUSE1:
            x = l->generic.left;
            y = l->generic.top;
            if( UI_CursorInRect( x, y, l->generic.right - x, l-
                                        >generic.bottom - y ) ) {
                index = (uis.cursory - y) / PROP_HEIGHT;
                l->oldvalue = l->curvalue;
                l->curvalue = index;

                if( l->generic.callback ) {
                    l->generic.callback( l, QM_ACTIVATED );
                    return menu_move_sound;
                }
            }
            return menu_null_sound;

        case K_KP_UPARROW:
        case K_UPARROW:
            l->oldvalue = l->curvalue;

            if( l->curvalue == 0 ) {
                l->curvalue = l->numitems - 1;
            }
            else {
                l->curvalue--;
            }
            return menu_move_sound;

        case K_KP_DOWNARROW:
        case K_DOWNARROW:
            l->oldvalue = l->curvalue;

            if( l->curvalue == l->numitems - 1 ) {
                l->curvalue = 0;;
            }
            else {
                l->curvalue++;
            }
            return menu_move_sound;
    }

    return Menu_DefaultKey( &ratePlayerMenuInfo.menu, key );
}

/*
=================
UI_RatePlayerMenu_ListDraw
=================
*/
static void UI_RatePlayerMenu_ListDraw( void *self ) {
    menulist_s  *l;
    int         x;
    int         y;
    int         i;
    float       *color;
    qboolean    hasfocus;
    int         style;

    l = (menulist_s *)self;

    hasfocus = (l->generic.parent->cursor == l->generic.menuPosition);

    x = 320;//l->generic.x;
    y = l->generic.y;
    for( i = 0; i < l->numitems; i++ ) {
        style = UI_LEFT|UI_SMALLFONT|UI_CENTER;
        if( i == l->curvalue ) {
            color = color_yellow;
            if( hasfocus ) {
                style |= UI_PULSE;
            }
        }
        else {
            color = color_orange;
        }

        UI_DrawProportionalString( x, y, l->itemnames[i], style, color
);
        y += PROP_HEIGHT;
    }
}

/*
===============
UI_RatePlayerMenu_ListEvent
===============
*/
static void UI_RatePlayerMenu_ListEvent( void *ptr, int event ) {
    int     id;
    int     selection;

    if (event != QM_ACTIVATED)
        return;

    id = ((menulist_s *)ptr)->generic.id;
    selection = ((menulist_s *)ptr)->curvalue;
    UI_PopMenu();
}

/*
===============
UI_RatePlayerMenu_BuildBotList
===============
*/
static void UI_RatePlayerMenu_BuildBotList( void ) {
    uiClientState_t cs;
    int     numPlayers;
    int     isBot;
    int     n;
    char    playerTeam;
    char    botTeam;
    char    info[MAX_INFO_STRING];

    for( n = 0; n < 9; n++ ) {
        ratePlayerMenuInfo.bots[n] = ratePlayerMenuInfo.botNames[n];
    }

    trap_GetClientState( &cs );

    Q_strncpyz( ratePlayerMenuInfo.botNames[0], "Everyone", 16 );
    ratePlayerMenuInfo.numBots = 1;

    trap_GetConfigString( CS_SERVERINFO, info, sizeof(info) );
    numPlayers = atoi( Info_ValueForKey( info, "sv_maxclients" ) );
    ratePlayerMenuInfo.gametype = atoi( Info_ValueForKey( info,
                                                    "g_gametype" ) );

    for( n = 0; n < numPlayers && ratePlayerMenuInfo.numBots < 9; n++ )
{
        trap_GetConfigString( CS_PLAYERS + n, info, MAX_INFO_STRING );

        if( n == cs.clientNum ) {
            playerTeam = *Info_ValueForKey( info, "t" );
            continue;
        }

        isBot = atoi( Info_ValueForKey( info, "skill" ) );
        if( !isBot ) {
            continue;
        }

        botTeam = *Info_ValueForKey( info, "t" );
        if( botTeam != playerTeam ) {
            continue;
        }

        Q_strncpyz(
           ratePlayerMenuInfo.botNames[ratePlayerMenuInfo.numBots],
           Info_ValueForKey( info, "n" ), 16 );
        Q_CleanStr(
           ratePlayerMenuInfo.botNames[ratePlayerMenuInfo.numBots] );
           ratePlayerMenuInfo.numBots++;
    }
}

/*
===============
UI_RatePlayerMenu_Init
===============
*/
static void UI_RatePlayerMenu_Init( void ) {
    int     y;

    UI_RatePlayerMenu_Cache();

    trap_R_RegisterShaderNoMip( ART_FRAME );
    trap_R_RegisterShaderNoMip( ART_BACK0 );
    trap_R_RegisterShaderNoMip( ART_BACK1 );

    memset( &ratePlayerMenuInfo, 0, sizeof(ratePlayerMenuInfo) );
    ratePlayerMenuInfo.menu.fullscreen = qfalse;
    ratePlayerMenuInfo.menu.key = UI_RatePlayerMenu_Key;

    UI_RatePlayerMenu_BuildBotList();

    ratePlayerMenuInfo.banner.generic.type      =          MTYPE_BTEXT;
    ratePlayerMenuInfo.banner.generic.x         =                  320;
    ratePlayerMenuInfo.banner.generic.y         =                   16;
    ratePlayerMenuInfo.banner.string            =        "RATE PLAYER";
    ratePlayerMenuInfo.banner.color             =          color_white;
    ratePlayerMenuInfo.banner.style             =            UI_CENTER;

    ratePlayerMenuInfo.frame.generic.type       =         MTYPE_BITMAP;
    ratePlayerMenuInfo.frame.generic.flags      =         QMF_INACTIVE;
    ratePlayerMenuInfo.frame.generic.name       =            ART_FRAME;
    ratePlayerMenuInfo.frame.generic.x          =              320-150;
    ratePlayerMenuInfo.frame.generic.y          =              240-100;
    ratePlayerMenuInfo.frame.width              =            300; //466
    ratePlayerMenuInfo.frame.height             =            160; //332

    ratePlayerMenuInfo.list.generic.type        =     MTYPE_SCROLLLIST;
    ratePlayerMenuInfo.list.generic.flags       =     QMF_PULSEIFFOCUS;
    ratePlayerMenuInfo.list.generic.ownerdraw   =
                                            UI_RatePlayerMenu_ListDraw;
    ratePlayerMenuInfo.list.generic.callback    =
                                           UI_RatePlayerMenu_ListEvent;
    ratePlayerMenuInfo.list.generic.x           =               320-64;
    ratePlayerMenuInfo.list.generic.y           =                  120;

    y = 170;
    ratePlayerMenuInfo.excellent.generic.type           =  MTYPE_PTEXT;
    ratePlayerMenuInfo.excellent.generic.flags          =
                                   QMF_CENTER_JUSTIFY|QMF_PULSEIFFOCUS;
    ratePlayerMenuInfo.excellent.generic.x              =          320;
    ratePlayerMenuInfo.excellent.generic.y              =            y;
    ratePlayerMenuInfo.excellent.generic.id             = ID_EXCELLENT;
    ratePlayerMenuInfo.excellent.generic.callback       =
                                                      RatePlayer_Event;
    ratePlayerMenuInfo.excellent.string                 =  "EXCELLENT";
ratePlayerMenuInfo.excellent.color                  =    color_red;

ratePlayerMenuInfo.excellent.style                  =
                                           UI_CENTER|UI_SMALLFONT;

y += RATE_PLAYER_MENU_VERTICAL_SPACING;
ratePlayerMenuInfo.average.generic.type             = MTYPE_PTEXT;
ratePlayerMenuInfo.average.generic.flags            =
                              QMF_CENTER_JUSTIFY|QMF_PULSEIFFOCUS;
ratePlayerMenuInfo.average.generic.x                =         320;
ratePlayerMenuInfo.average.generic.y                =           y;
ratePlayerMenuInfo.average.generic.id               =  ID_AVERAGE;
ratePlayerMenuInfo.average.generic.callback         =
                                                 RatePlayer_Event;
ratePlayerMenuInfo.average.string                   =   "AVERAGE";
ratePlayerMenuInfo.average.color                    =   color_red;
ratePlayerMenuInfo.average.style                    =
                                           UI_CENTER|UI_SMALLFONT;

y += RATE_PLAYER_MENU_VERTICAL_SPACING;
ratePlayerMenuInfo.belowAverage.generic.type        =
                                                      MTYPE_PTEXT;
ratePlayerMenuInfo.belowAverage.generic.flags       =
                              QMF_CENTER_JUSTIFY|QMF_PULSEIFFOCUS;
ratePlayerMenuInfo.belowAverage.generic.x           =         320;
ratePlayerMenuInfo.belowAverage.generic.y           =           y;
ratePlayerMenuInfo.belowAverage.generic.id          =
                                                 ID_BELOW_AVERAGE;
ratePlayerMenuInfo.belowAverage.generic.callback    =
                                                 RatePlayer_Event;
ratePlayerMenuInfo.belowAverage.string              =
                                                  "BELOW AVERAGE";
ratePlayerMenuInfo.belowAverage.color               =   color_red;
      ratePlayerMenuInfo.belowAverage.style         =
                                           UI_CENTER|UI_SMALLFONT;

y += RATE_PLAYER_MENU_VERTICAL_SPACING;
ratePlayerMenuInfo.poor.generic.type                =
                                                      MTYPE_PTEXT;
      ratePlayerMenuInfo.poor.generic.flags         =

  QMF_CENTER_JUSTIFY|QMF_PULSEIFFOCUS;
ratePlayerMenuInfo.poor.generic.x                   =         320;
ratePlayerMenuInfo.poor.generic.y                   =           y;
ratePlayerMenuInfo.poor.generic.id                  =     ID_POOR;
ratePlayerMenuInfo.poor.generic.callback        =

                      RatePlayer_Event;

ratePlayerMenuInfo.poor.string                      =      "POOR";
ratePlayerMenuInfo.poor.color                       =   color_red;
ratePlayerMenuInfo.poor.style                       =
                                           UI_CENTER|UI_SMALLFONT;

ratePlayerMenuInfo.back.generic.type                =MTYPE_BITMAP;
ratePlayerMenuInfo.back.generic.name                =   ART_BACK0;
```

```
    ratePlayerMenuInfo.back.generic.flags                =
                                    QMF_LEFT_JUSTIFY|QMF_PULSEIFFOCUS;
    ratePlayerMenuInfo.back.generic.callback             =
                                          UI_RatePlayerMenu_BackEvent;
    ratePlayerMenuInfo.back.generic.x                    =          0;
    ratePlayerMenuInfo.back.generic.y                    =     480-64;
    ratePlayerMenuInfo.back.width                        =        128;
    ratePlayerMenuInfo.back.height                       =         64;
    ratePlayerMenuInfo.back.focuspic                     =  ART_BACK1;

    Menu_AddItem( &ratePlayerMenuInfo.menu, &ratePlayerMenuInfo.banner
                                                                     );
    Menu_AddItem( &ratePlayerMenuInfo.menu, &ratePlayerMenuInfo.frame
                                                                     );
    Menu_AddItem( &ratePlayerMenuInfo.menu, &ratePlayerMenuInfo.list );
    Menu_AddItem( &ratePlayerMenuInfo.menu, &ratePlayerMenuInfo.back );
    Menu_AddItem( &ratePlayerMenuInfo.menu,
                                       &ratePlayerMenuInfo.excellent );
    Menu_AddItem( &ratePlayerMenuInfo.menu, &ratePlayerMenuInfo.average
                                                                     );
    Menu_AddItem( &ratePlayerMenuInfo.menu,
                                    &ratePlayerMenuInfo.belowAverage );
    Menu_AddItem( &ratePlayerMenuInfo.menu, &ratePlayerMenuInfo.poor );

    ratePlayerMenuInfo.list.generic.left                 =         220;
    ratePlayerMenuInfo.list.generic.top                  =
                                     ratePlayerMenuInfo.list.generic.y;
    ratePlayerMenuInfo.list.generic.right                =         420;
//  UI_RatePlayerMenu_SetList( ID_LIST_BOTS );
}

/*
=================
UI_RatePlayerMenu_Cache
=================
*/
void UI_RatePlayerMenu_Cache( void ) {
    trap_R_RegisterShaderNoMip( ART_FRAME );
    trap_R_RegisterShaderNoMip( ART_BACK0 );
    trap_R_RegisterShaderNoMip( ART_BACK1 );
}

/*
===============
UI_RatePlayerMenu
===============
*/
void UI_RatePlayerMenu( void ) {
    UI_RatePlayerMenu_Init();
    UI_PushMenu( &ratePlayerMenuInfo.menu );
}

/*
===============
UI_RatePlayerMenu_f
===============
*/
```

```
void UI_RatePlayerMenu_f( void ) {
    uiClientState_t cs;
    char    info[MAX_INFO_STRING];
    int     team;

    // make sure it's a team game
    trap_GetConfigString( CS_SERVERINFO, info, sizeof(info) );
    ratePlayerMenuInfo.gametype = atoi( Info_ValueForKey( info,
                                                    "g_gametype" ) );
    if( ratePlayerMenuInfo.gametype < GT_TEAM ) {
        return;
    }

    // not available to spectators
    trap_GetClientState( &cs );
    trap_GetConfigString( CS_PLAYERS + cs.clientNum, info,
                                                   MAX_INFO_STRING );
    team = atoi( Info_ValueForKey( info, "t" ) );
    if( team == TEAM_SPECTATOR ) {
        return;
    }

    UI_RatePlayerMenu();
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Forsyth, D. (1990). Group Structure. Group Dynamics. Retrieved April 21, 2000 from the World Wide Web: http://www.vcu.edu/hasweb/psy/psy633/read.html#amer

Foushee, H. (1984). Dyads and triads at 35,000 feet: factors affecting group process and aircrew performance [Abstract]. American Psychologist, 39, 885-893. Retrieved April 21, 2000 from the World Wide Web: http://www.vcu.edu/hasweb/psy/psy633/foushee.htm

Knight, C., & Munro, M. (1998). Using an existing game engine to facilitate multi-user information visualization. Retrieved May 15, 2000 from the World Wide Web: http://www.dur.ac.uk/~dcs1crk/workfiles/documents/multi-user_software_vis/multi-user_software_vis.html

McCraw, R., & Bearden, D. (1990). Personality factors in failure to adapt to the military [Abstract]. Military Medicine, 155(3), 127-130. Retrieved April 24, 2000 from the PsycINFO database on the World Wide Web: http://psycinfo.apa.org

McDonald, B. (1999). Bot Weapon/Itemweights Editing Guide. Retrieved August 18, 2000 from the World Wide Web: http://www.planetquake.com/quake3/q3aguide/bot-tutorial_intro.shtml

Miliano, V. (1999). Application of a 3D game engine to enhance the design, visualization and presentation of commercial real estate. Retrieved June 23, 2000 from the World Wide Web: http://www.unrealty.net/vsmm99/

National Research Council. (1997). Modeling and Simulation: Linking Entertainment and Defense. Washington D.C.: National Academy Press.

Singhal, S., & Zyda, M. (1999). Networked Virtual Environments: Design and Implementation. New York: Addison-Wesley Company, Inc.

Vince, J. (1992). 3-D Computer Animation. New York: Addison-Wesley Publishing Company Inc.

Weiner, H. (1990). Group-level and individual-level mediators of the relationship between soldier satisfaction with social support and performance motivation [Abstract]. Military Psychology, 2(1), 21-32. Retrieved April 24, 2000 from PsycINFO database on the World Wide Web: http://psycinfo.apa.org

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
8725 John J. Kingman Road, Ste 0944
Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library 2
Naval Postgraduate School
411 Dyer Rd.
Monterey, California 93943-5101

3. RADM Richard Mayo 1
CNO, N6
2000 Navy Pentagon
Washington, DC 20350-2000

3. CDR George Phillips, USN (Ret) 1
CNO, N6M1
2000 Navy Pentagon
Washington, DC 20350-2000

4. Dr. Allen Zeman 1
CNO, N7B
2000 Navy Pentagon
Washington, DC 20350-2000

5. Dr. Michael Macedonia 1
Chief Scientist and Technical Director
US Army STRICOM
12350 Research Parkway
Orlando, FL 32826-3276

6. Dr. Casey Wardynski 2
United States Military Academy
Office of Economic & Manpower Analysis
607 Cullum Rd, Floor 1B, Rm B109
West Point, NY 10996-1798

7. Dr. Michael Capps 3
Computer Science Department
Naval Postgraduate School
Monterey, CA 93940-5000

8. LT Jeffrey D. DeBrine 1
925 Langdon Ct.
Annapolis, MD 21403

9. LT Donald E. Morrow C/O Mike Lewis 1
115 E. Williams St.
Broadwell, IL 62623

72 290NPG 3034
TH
6/02 22527-200 NLB

DUDLEY KNOX LIBRARY
3 2768 00403097 3